Algorithms

**Homework 3**

**Instructor:** Anna Karlin

**Due February 6, by 11:59pm**

**Instructions:**

Include pseudo-code for your algorithms, a run-time analysis and a proof of correctness.

Also, read these instructions carefully: [http://courses.cs.washington.edu/courses/cse417/16wi/grading.html](http://courses.cs.washington.edu/courses/cse417/16wi/grading.html)

**Instructions forthcoming on how to turn the homework in.**

1. Solve these problems on canvas: [https://canvas.uw.edu/courses/1021503/quizzes/878962](https://canvas.uw.edu/courses/1021503/quizzes/878962)

2. Often there are multiple shortest paths between two nodes. Give a linear time algorithm for the following task:

   **Input:** An undirected graph \( G = (V, E) \) with unit edge lengths; nodes \( u, v \in V \).
   
   **Output:** The number of distinct shortest paths from \( u \) to \( v \).

3. Consider a directed graph on \( n \) vertices, where each vertex has exactly one outgoing edge. This graph consists of a collection of cycles as well as additional vertices that have paths to the cycles, which we will call the branches. We define the *weight* of the cycle to be the total number of vertices that are either on the cycle or on branches that are connected to the cycle.

   Give pseudocode for a linear time algorithm that identifies all of the cycles and computes the length and weight of each cycle.

4. Implement your algorithm from the previous problem for finding the cycles in an out-degree one graph. Your algorithm should be designed to work on very large graphs, e.g., with \( n = 100,000,000 \).

   Write an input generator which creates completely random out-degree one graphs where each vertex points to another vertex chosen uniformly at random and run your program on inputs from your generator of various sizes.

   You are free to write in any programming language you like. The quality of your algorithm may be graded, but the actual quality of the code will not be graded. The expectation is that you write the algorithmic code yourself – but you can use other code or libraries for supporting operations.

   We will ask you to turn in your algorithmic code, including the code generating the random inputs.

   Also please write up an answer to the following question: As the size of the problem increases – how does the number of cycles, and the length and the weight of the cycles change, when the input is a random graph with out-degree one? Provide data to support your answer.

5. Problem 3.15 on page 102 of [DPV].
6. We have a connected, undirected graph \( G = (V, E) \), and a specific vertex \( u \in V \). Suppose that we compute a depth-first search tree rooted at \( u \), and obtain a tree \( T \) that includes all nodes of \( G \). Suppose we then compute a breadth-first search tree rooted at \( u \), and obtain the same tree \( T \). Prove that \( G = T \). (In other words, if \( T \) is both a depth-first search tree and a breadth-first search tree rooted at \( u \), then \( G \) cannot contain any edges that do not belong to \( T \).)

7. I may be adding one or two more problems on 1/30. (They will depend on how far we get in the material before that.)